Faking a State Resource in EUROPA

While we hope to get a truly generic version of state resources included in EUROPA (see Possible New Features at bottom of this page), this page describes:

1. How a unary resource can be faked easily using the existing framework, and
2. How profiles can be subclassed to get different behavior.

Overview

For a description of how we fake a unary state resource (ie a resource that can be 'on' or 'off' and has certain things that can require it to be on), see the comments in ExampleStateResourceCustomCode.hh.

Example Implementation

A new project automatically includes its own module that is loaded into the EUROPA engine and stub C++ files to hold custom code (see makeproject). Therefore, extending EUROPA components simply requires C++ code be added to those existing files. Consider the ExampleStateResource project here, which can be downloaded with:

```
svn co https://babelfish.arc.nasa.gov/svn/europa/benchmarks/trunk/ExampleStateResource  ./
```

We have added an 'ExampleConstraint' that restricts a variable to have integer bounds. To create and use the constraint involved these steps:

1. Declare and define the 'StateProfile?' in ExampleStateResourceCustomCode.hh and ExampleStateResourceCustomCode.cc.
2. Register the StateProfile. In ModuleExampleStateResource.cc:

```
        FactoryMgr* pfm = (FactoryMgr*)engine->getComponent("ProfileFactoryMgr");
        REGISTER_PROFILE(pfm, StateProfile, StateProfile);
```

3. Create a StateResource class in NDDL that extends Reservoir, and, most importantly, sets 'profileType' to 'StateProfile?'. In ExampleStateResource-model.nddl, notice:

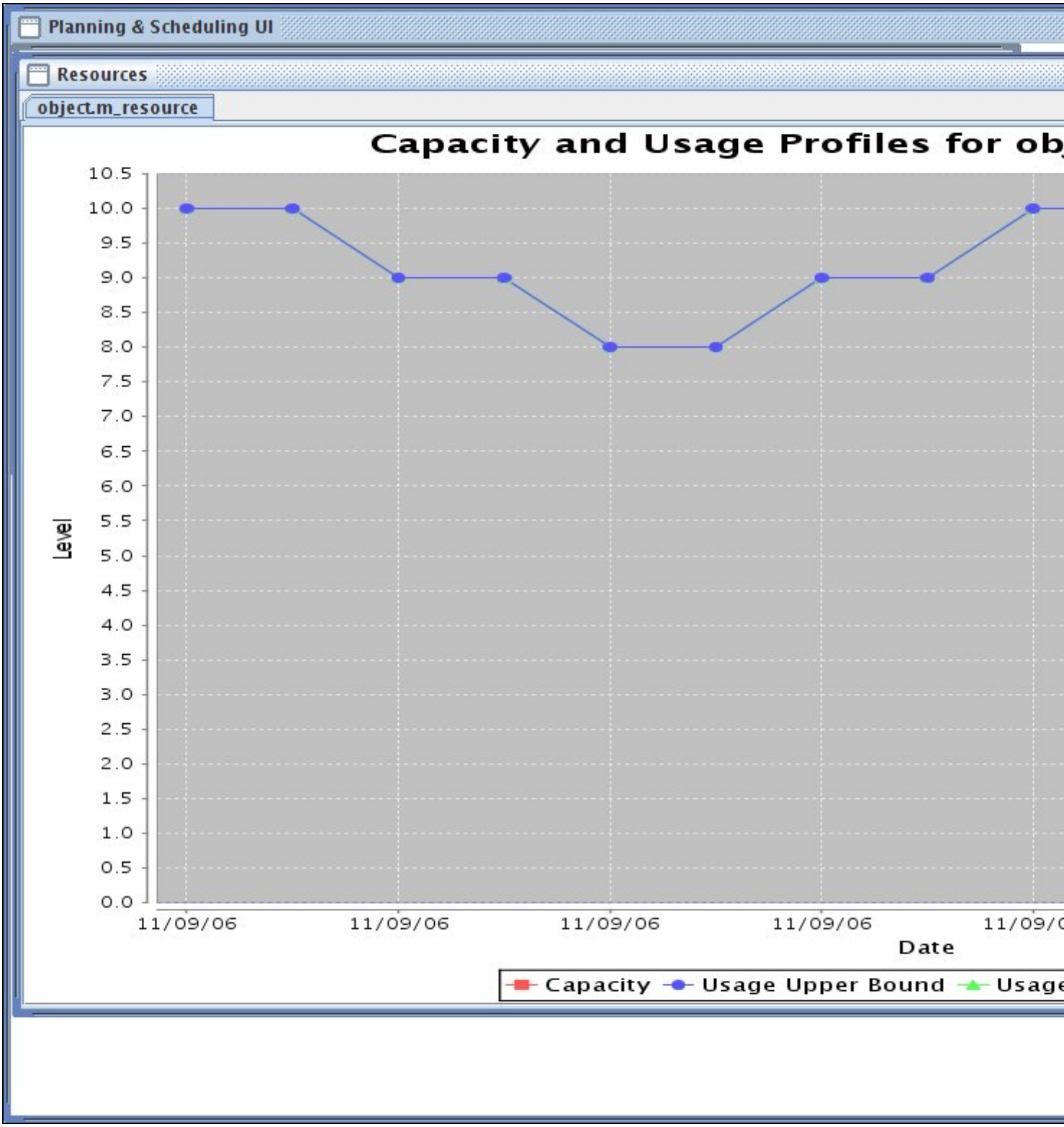
```
class StateResource extends Reservoir {
     string profileType;

     StateResource()
     {
             super(0.0 ,0.0, 10.0); // initial, lower_limit, upper_limit
             profileType = "StateProfile";
     }
}
```

1. Create a class that contains the StateResource and has predicates to turn it off and on, as well as a predicate to indicate the state is required. See ExampleStateResource-model.nddl.
2. Create an example initial state to illustrate the desired behavior. See ExampleStateResource-initial-state.nddl.

NOTES:

1. To get the desired behavior, the quantity used for 'STATE_COND_TRUE' in ExampleStateResourceCustomCode.cc must match the quantity consumed and produced by the turnOn and turnOff predicates in ExampleStateResource-model.nddl. We use 10 for each to illustrate behavior, but in practice a much bigger number should be used; it must be larger than the total number of tokens at any given time that might require the state to be on.
2. To see the results of the above example, run 'ant.' You will see the following profile, which shows:
    1. The 'turnOn' at time 0 makes the level start at 10.
    2. A second 'turnOn' at time 1 makes no difference.
    3. Three 'require' tokens make the level dip below 10, while they require the state.
    4. A 'turnOff' at time 10 makes the level drop to 0. If there was still a 'require' at this point, there would be a flaw, and the solver would fail to find a solution.
    5. A second 'turnOff' at time 12 makes no difference.
    6. A final 'turnOn' at time 14 sets the level back to 10.


Planning & Scheduling UI
Resources
object.m_resource
Capacity and Usage Profiles for ob
10.5
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Level
11/09/06
11/09/06
11/09/06
11/09/06
11/09/0
Date
Capacity
Usage Upper Bound
Usage